RoboSim for Integrated Computing and STEM Education

Mr. Kevin James Gucwa, University of California, Davis

Kevin Gucwa is a PhD candidate in Mechanical Engineering at the University of California at Davis. He is studying computer simulations in relation to modular robotics and their applications to K-12 STEM education. Since 2010 he has been involved with the UC Davis Center for Integrated Computing and STEM Education. This organization is working to study and improve K-12 STEM Education through the integration of computing and robotics into STEM classes to increase student engagement.

Prof. Harry H. Cheng, University of California, Davis

Harry H. Cheng is a Professor in the Department of Mechanical and Aerospace Engineering, Graduate Group in Computer Science, and Graduate Group in Education at the University of California, Davis, where he is also the Director of the UC Davis Center for Integrated Computing and STEM Education (http://c-stem.ucdavis.edu) and Director of the Integration Engineering Laboratory. His current research includes developing computing and robotics technologies and integrate them into STEM education in both formal and informal settings for integrated learning. From 1989 to 1992, he was a Senior Engineer for robotic automation systems with the Research and Development Division, United Parcel Service. He has authored and coauthored more than 170 papers in refereed journals and conference proceedings. He holds two U.S. patents. He is the author of the book ”C for Engineers and Scientists: An Interpretive Approach” (McGraw-Hill, 2009). He is the co-founder of SoftIntegration, Inc. and Barobo, Inc. He received a M.S. degree in mathematics and a Ph.D. degree in mechanical engineering from the University of Illinois at Chicago in 1986 and 1989, respectively. He is a Fellow of the American Society of Mechanical Engineers and a Senior Member of IEEE. Dr. Cheng received the ASME’s MESA Achievement Award for a cumulative contribution to the field of Mechatronic and Embedded Systems and Applications, a Research Initiation Award from the National Science Foundation, the Best Paper Award and Best Student Paper Award at the IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, the Procter and Gamble Best Paper Award as well as the Waldron Award at the Applied Mechanisms and Robotics Conference. He received an Outstanding Contribution Award from United Parcel Service, Inc. He was the General Chair of the 2009 ASME/ IEEE International Conference on Mechatronic and Embedded Systems and Applications and the Program Chair of the 2006 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications.

RoboSim for Integrated Computing and STEM Education

Abstract

This paper describes the design, implementation, and application of RoboSim, a robot virtual environment, for integrated computing and STEM education in K-12 schools. Robots are being increasingly used in schools for hands-on project-based learning and motivating students to pursue careers in Science, Technology, Engineering, and Mathematics (STEM). However high costs and hardware issues are often prohibitive for using robotics as often as desired in mathematics and science classroom teaching. Due to the tight schedule for teaching math and science subjects, hardware mishap and failure, such as battery charging, disconnection, and breaking will have serious negative impact on classroom teaching and learning. Since physical robots are necessary for hands-on learning, a well-designed robot virtual environment can provide a time and cost effective complement for the physical robots when teaching STEM subjects. Integrated computing and STEM education integrates computing with the robotics technology into STEM education to increase student interest and help them learn the STEM subjects. Mobot and Linkbot are reconfigurable modular robots, designed especially for hands-on project-based integrated learning of computing and STEM subjects. While each module is a fully functional robot, they can be easily connected with snapped connectors to form various geometric configurations for myriad applications. Both Mobot and Linkbot can be conveniently controlled with a user-friendly C/C++ interpreter Ch. They have been successfully used in classroom teaching of STEM subjects including Algebra. RoboSim is a robot virtual environment for Mobot and Linkbot. Code developed with RoboSim can also be used to control the hardware robots without any modification. It allows students to program robots without physical access to the hardware and provides idealized validation of both the hardware robot motion and materials being taught. A virtual environment without disturbance and hardware failure often allows more effective teaching and learning of subject matters. Feedback from both instructors and students about RoboSim during initial field testing is quite positive indicating that it is an effective tool for integrated learning of computing and STEM subjects.

1 Introduction

Since it was first mandated in 2004, Algebra has been a requirement for graduation from high school in the state of California.[1] Without passing Algebra students are not able to move onto higher mathematics classes which form the basis of a career within the STEM

fields. A study by the California Dropout Research Project[2] found that students within the Los Angeles Unified School District who passed Algebra 1 by their freshman year of high school were twice as likely to graduate with rates of 70% and 35%, respectively. Enrollment in STEM degree fields at US Universities has increased from 519,000 students in 1994-1995 to 578,000 students in 2003-2004. However, the proportion of undergraduate degrees in STEM fields declined from 32% to 27% as a percentage of all degrees.[3]

Hands-on learning and its benefits have been looked at for years as a possible way to engage students and help them learn more effectively than traditional teaching methods.[4,5] Rutherford[6] stated in 1993 that hands-on learning is a powerful idea and that engaging students actively and thoughtfully in their studies pays off in better learning. Presently the prevalence of hands-on learning activities within classrooms is greatly increasing due to these benefits. Adding these activities to Algebra classrooms is important as it is the basis of most STEM classes and students need to fully understand the material.

Educational robots are becoming an increasingly popular method of implementing hands-on activities within classrooms. Lego Mindstorms is the most utilized classroom robotics kit but it is not designed for integration directly into Algebra classrooms. The number of pieces and reconfigurability are great for stimulating creative thinking but not conducive to classrooms with thirty students. Having simple robots which convey the educational concepts allows learning to happen while exploring math rather than robotics. The Linkbot and Mobot are modular robots which have been designed as simple, easy-to-use classroom robots geared toward math education first and foremost. Each individual robot can be utilized as a two-wheel vehicle with set-up happening by snapping on wheels and a caster or in more complicated configurations.

The use of simulation as a design tool for new robotic designs is increasing steadily due to the cost effective and efficient prototyping which can happen. Before robots are built and tested, the designs can be verified within the environment which they will ultimately be operating. Many general purpose simulators have been developed which allow for simulated robot motion. Gazebo[7] is one of the most popular which integrates into the Player[8] project to allow code to be put onto the hardware with only minor modifications. Others, such as V-REP,[9] XPERSim,[10] YARS,[11] and Marilou,[12] all provide a simulation environment where robotic models can be created and tested. The design and use of these general simulators is geared at researchers who wish to test new robot hardware designs or code which is to operate on autonomous robots. There is no consideration put forth to make them easy to use for other types of learning.

The benefits of robots in the classroom are partially offset by the issues which arise in dealing with hardware. Battery charging and disconnection from computers must be dealt with in addition to a classroom full of students and robots. These issues can be addressed by using virtual robots on each student's computer. Despite the work being put into developing easy to use robots for hands-on learning, very little has been done to combine computer simulations with the robots. The Robot Virtual Worlds[13] program developed by Carnegie Mellon's Robotics Academy is a cross between a video game and educational lessons. By eliminating the clean-up associated with hardware and allowing students to move at their own pace, they[14] found that the virtual robots were a more effective

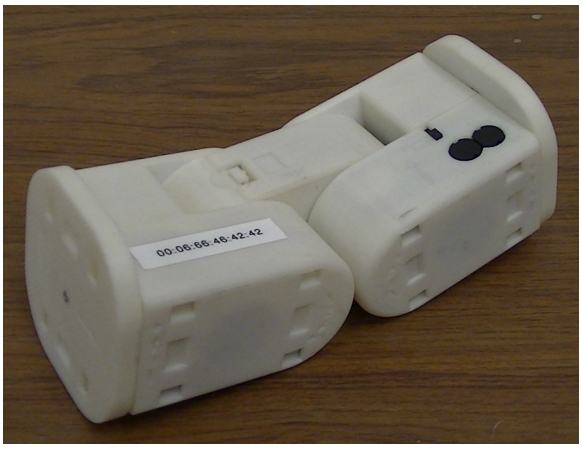

Figure 1: Mobot

introduction to robotics for classrooms of students than the hardware robots. After the introduction, all students progressed to the hardware robots for hands-on learning. The presence of the virtual robots cannot take the place of hands-on learning within the classroom but when interspersed, it can be effective at quickly introducing students to the robot concepts.

RoboSim is designed to allow the advantages of simulation of robotic motions to be applied to mathematics and science teaching within the classroom. Bypassing the hardware eliminates numerous issues when trying to work with a full class of students. The easy-to-use RoboSim and Ch environment are effective at allowing students to concentrate on the topic and use the robots as a supplement to the lesson.

2 Reconfigurable Robots Mobot and Linkbot

The modular robots Mobot and Linkbot,[15] developed by Barobo, Inc., are educational versions of the iMobot[16,17] designed in the Integration Engineering Lab at the University of California, Davis. The iMobot was designed to enhance the characteristics of previously designed modular robots with greater simplicity and mobility.

The Mobot module is a lower cost and easy to control version with applications in education. The primary focus of the Mobot within education is mathematics classrooms, specifically Algebra I, where students can benefit from a hands-on experience to supplement their classroom activities. A small number of pieces and a lower cost allows sets of robots to be used within class to help visualize mathematic and scientific topics. The Mobot has four, simple, rotational degrees-of-freedom, as shown in Figure 1. The two main bodies of the Mobot provide the primary means of configuration while the two end bodies, called faceplates, allow it to roll around.

The Linkbot modular robot is a simpler educational robot than the Mobot with less

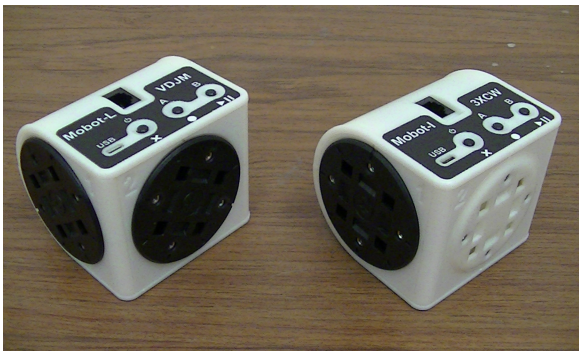

Figure 2: Linkbot-L and Linkbot-I

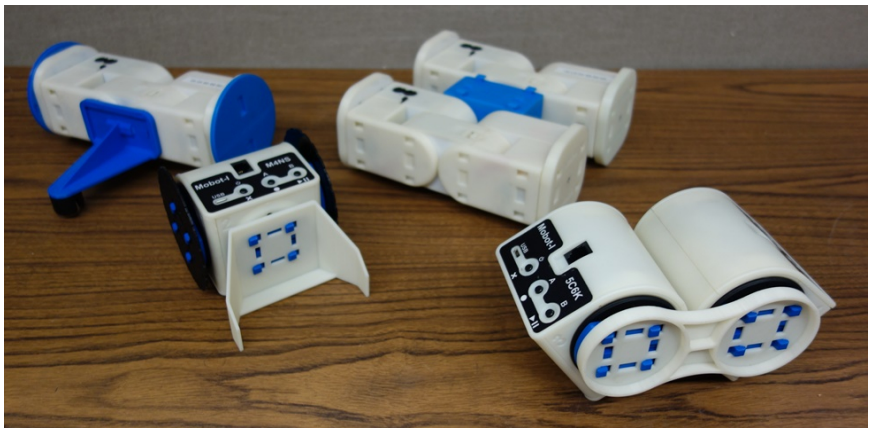

Figure 3: Mobot accessories.

degrees of freedom but an even lower cost. The capabilities of the Mobot, given its four degrees-of-freedom, has at times been too much for the lessons within an Algebra classroom. Thus the Linkbot improves upon the Mobot design for classroom use by removing unnecessary complexity. There are two versions of the Linkbot each of which have two rotational joints. Shown on the left of Figure 2, the Linkbot-L has two joints, shown in black, next to each other. The Linkbot-I, shown on the right side of Figure 2, has the two joints on opposite sides of the robot. As is often necessary for Algebra, by adding wheels and a caster the Linkbot-I can become a simple two-wheel drive robot.

Adapting the capabilities of the iMobot to education involved reducing the complexity and cost of each module. The resulting system is made up of self-contained robot modules with snapping connectors for easy construction of novel configurations. Shown in Figure 3, the accessories for the Mobot and Linkbot include wheels, casters, and a variety of others which facilitate rolling and walking configurations. All accessories are designed to enhance the capabilities of the robots by allowing them to be used for myriad applications.

One of the unique capabilities of the Mobot and Linkbot system is the ability to program controlling code for the robots through Ch,[18] a C/C++ interpreter which is described in more detail later. From within Ch students can use the simple set of functions available[19] to connect to robots, move them around, and collect data from them. Code is not

downloaded to the robots each time a change needs to be made. The robots are remotely controlled by the Ch code and command are sent wirelessly to the robots. Recording data about the robot's motions was implemented to provide a means of applying the robots as a verification tool for mathematical expressions within a classroom. Recorded data is plotted to show that the motion of the robots follow the mathematical function used to guide their motion.

3 C/C++ Interpreter Ch

Ch provides an ideal environment for programming robots especially in an educational setting. A language used for robots must be easy to learn but provide capabilities for advanced users to use the full power of the system. Ch is an interpreted C/C++ language that implements most of the standard C and C++ features[20,21] with additional items useful for mathematics and engineering such as complex numbers[22] and plotting.[23]

ChIDE is an Integrated Development Environment to easily write and run interpreted code. It provides a simple interface which facilitates debugging by allowing students to step through code one line at a time and view values of variables at that time. Code run within Ch through ChIDE display easily to understood error messages, which include a line number where the error most likely occurred, to help the user quickly debug the problem. Therefore, it reduces the time which students are waiting for help.

4 Design and Implementation of RoboSim

To be able to take advantage of the features Ch brings to the classroom, RoboSim was designed to integrate completely into this interpreted environment. To interact with the virtual robots, two pieces of software are required. The RoboSim Graphical User Interface (GUI) allows students to configure the virtual world before execution from within ChIDE.

4.1 RoboSim Configuration

Initial configuration of the virtual environment is necessary to effectively utilize its capabilities. A simple configuration dialog has been created to allow students to set up the necessary variables to run a simulation. The options have been simplified to mimic what must be done before running the hardware robots. The GUI, shown in Figure 4, presents two sections to the student. First an option to allow switching between the hardware robots and the virtual ones. By clicking either option, the unmodified code from within Ch will either be sent to the hardware robots or display the virtual robots. To configure the positioning of the virtual robots, the remainder of the GUI is set up to allow easy adjustments to the starting locations.

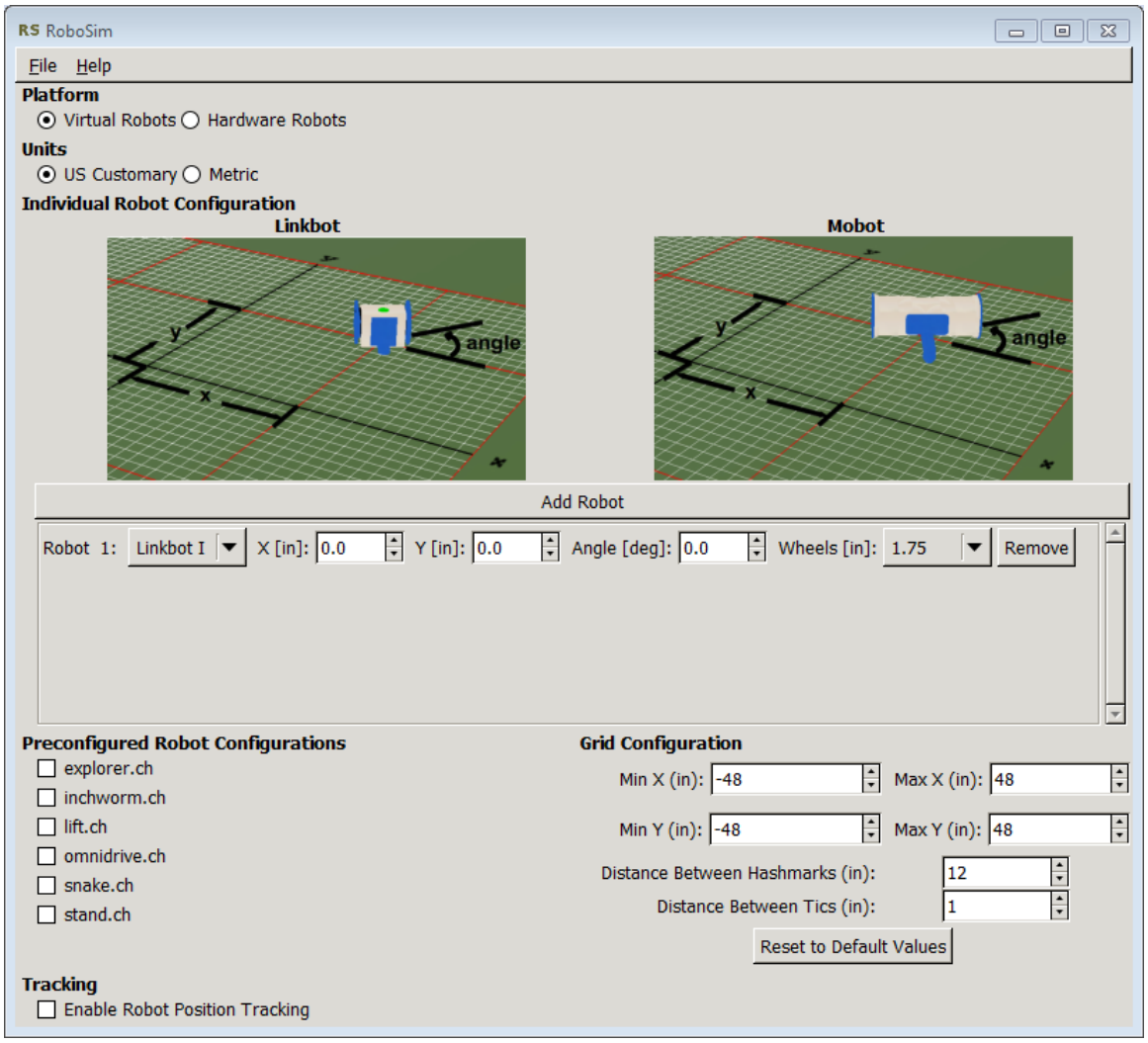

Figure 4: The RoboSim GUI.

To be most useful in mathematics classrooms, RoboSim was designed to be easy to use for the most common Algebra topics. Number lines and linear equations are common topics which have been enhanced by the Linkbots and Mobots. Typically the robots are utilized as two wheeled vehicles to move around on number mats. Given these use cases, the interface to RoboSim has been designed to quickly let students set up their virtual robots to mimic the positioning of the hardware ones. Figure 5 shows the main robot configuration dialog for RoboSim. Each robot can be added to the scene one at a time and its initial properties can be set. X and Y offsets from the origin are specified as well as the angle of the robot from the X-axis. With these three pieces of information, a plethora of starting configurations can be set up to control single or multiple robots. To investigate the influence of wheel diameter on motion, three default wheel sizes in addition to user-specified custom wheel sizes are available through a drop down menu. Adding the wheels turns the robots into a stable, miniature vehicle for rolling around.

To be able to see how far robots have moved, a grid is enabled under the robots. There are

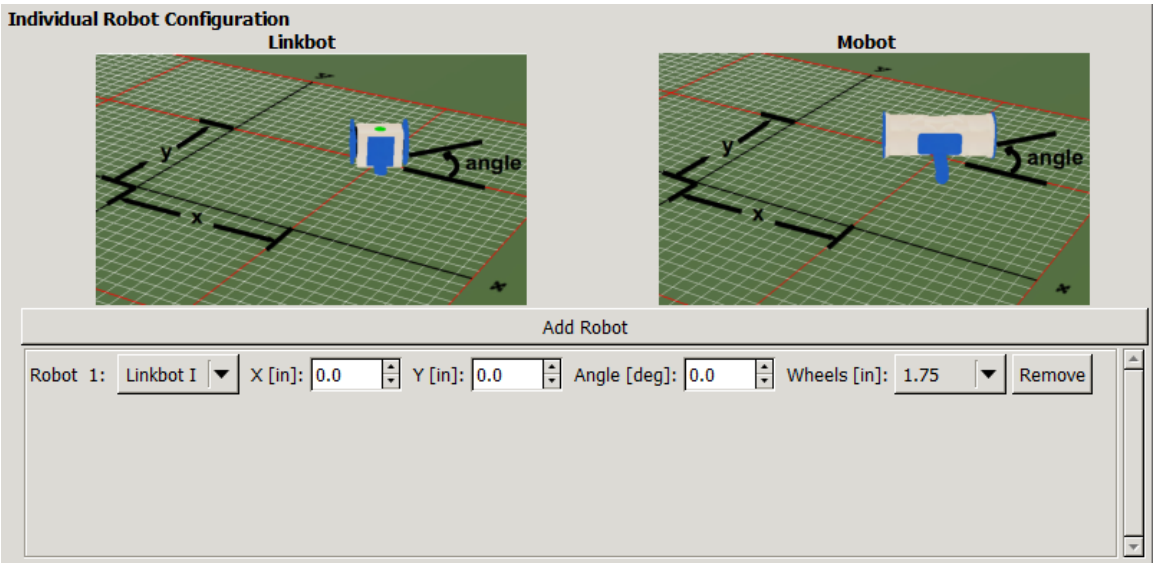

Figure 5: Individual robot configuration dialog.

six options to alter the layout of the grid lines; minimum and maximum X and Y locations, distance between hash marks, and distance between tics. The defaults are set up with tics at each inch, hash marks every foot, and a total of four feet of grids in both X and Y directions. Switching between the US customary and metric units changes the grid layout from inches to centimeters. Moving around on these grids makes visualization of many mathematics concepts much easier by allowing confirmation that when the robot moves twelve inches it lands on top of the line at one foot.

4.2 ChIDE and RoboSim Scene

Once RoboSim has been configured, the code written within ChIDE does not have to be modified to show the virtual robots. Running the code will generate a pop-up window, called RoboSim Scene, which shows the virtual robots moving about the grid which was configured within the RoboSim GUI. To align with the mathematics activities for which this is geared, it is necessary to know how far a robot has moved and where it is right now within the grid. The advantage of virtual robots over hardware robots is that it is possible to know exactly where all robots are at all times. Since the positions of robots are always known, they can be tracked from within the simulation. It is possible to have a line following the robots to display the path along which the robot has moved. Once a simulation is running, clicking on each robot will generate a pop-up message giving information about the robot number and its current position. The tracking line can be enabled or disabled through the keyboard shortcuts within the simulation. These two features can be seen in Figure 6. It is also possible to hide the robots from view and only see the tracking line curves as they are drawn onto the RoboSim Scene.

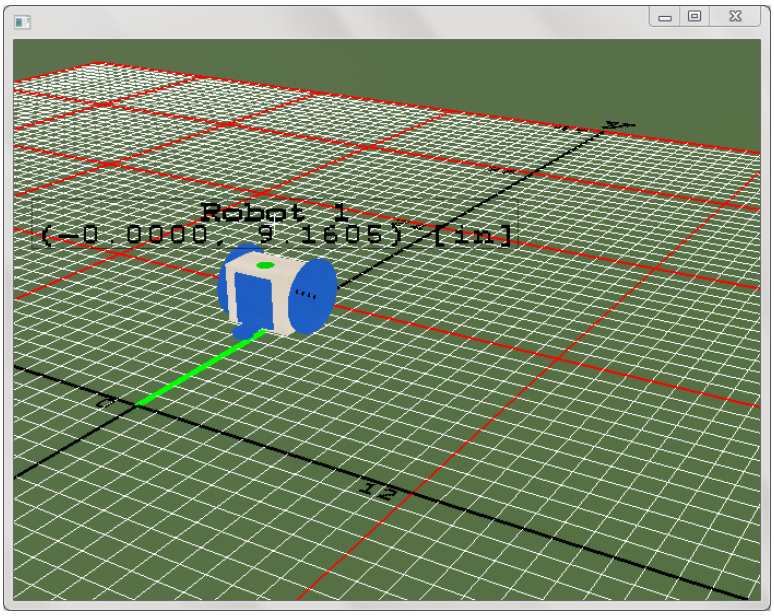

Figure 6: The RoboSim Scene with pop-up showing current robot position.

5 Programming Mobot and Linkbot with Ch

Programming through ChIDE to control the Mobot and Linkbot modular robots is greatly simplified by the simple set of functions available to control the robots. Each robot is created within the program and each function available allows joints to be moved as necessary.

Teaching students to code is important to help them become aware of the computer programming which is the underpinning of so many jobs. The introduction to programming should be simple and easy to understand and not get in the way of the mathematics. The programming environment set up through ChIDE and RoboSim allows teachers to quickly introduce how to program the robots to move from one location to another. The functions are straight-forward enough to easily be introduced into many classroom environments but also show the true C programming syntax utilized by computer scientists daily.

The ability to quickly move the robots around is essential to maintaining student interest. One of the basic functions available to students is the ability to move the robots forward or backward along a straight line.

```
robot.moveDistance(distance, radius);
```

The function needs two arguments; the distance to move (`distance`) and the radius of the wheels (`radius`) attached to the robot. From this information the angle needed to turn the wheels is calculated and used to move the robot forward. While simple to program, there are a few mathematics concepts which can be talked about while watching a robot move the specified distance.

In addition to moving in a straight line, the robots have the ability to easily turn left and

right facilitating the creation of geometric shapes.

```
robot.turnLeft(angle, radius, trackwidth);
robot.turnRight(angle, radius, trackwidth);
```

Each turn command needs three arguments; the number of degrees to turn (`angle`), the wheel radius (`radius`), and the distance between the wheels of the robot (`trackwidth`). Taking this information, the robot calculates how far each wheel needs to turn.

In addition to the basic functions, dozens more are available to get information from the robot, move joints to different positions, and move the entire robot in a coordinated manner. A subset of the available functions are listed below to show the options available for controlling the robots.

```
move() - move the robot's joints relative to the current angle
moveTo() - move the robot's joint to a specific angle
recordDistance() - record distance versus time of robot's motion
recordAngle() - record the robot's joint's angles over time
```

Combining the functions in sequence creates code which allows the robot to behave as a student desires to elucidate the mathematics being studied. In addition to the visual of the virtual robots, combining the recording functions with the Ch plotting capabilities, graphs of the robots motion can be generated and saved onto the computer.

5.1 Controlling a Single Robot in RoboSim

The most simple case to study is a single Mobot or Linkbot moving around as a two-wheeled vehicle simulating a point in space to trace a path. Taking the distance function mentioned above and combining it with a few lines of C++ code, a complete program, as shown in Program 1, is written which moves the robot forward.

```
#include <linkbot.h>

CLinkbotI robot;

double distance = 5, radius = 1.75;

robot.connect();
robot.moveDistance(distance, radius);
```

Program 1: Single robot control code.

Only five lines of code are necessary to connect to the robot and move it forward. The first line is C++ syntax to allow the code to know about all of the functions available to control the robot. The second line creates the robot within the code so that the functions can interact with the correct robot. Variables are created to store the distance to move and the radius of the robot's wheels. Using these well-named variables throughout the code allows students to see what information they are passing to the robot. The last two lines tell the

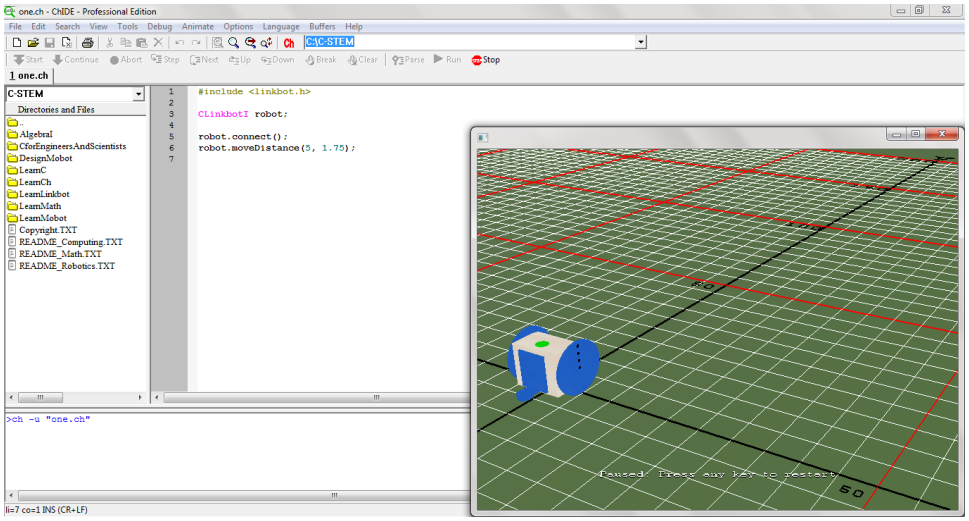

Figure 7: Single robot running in RoboSim through ChIDE.

computer to connect to the robot and move forward. As long as the two values given to the `moveDistance` function are in the same units, such as inches, centimeters, or any other unit of measure being taught, the robot will move correctly.

To configure RoboSim to run this code, the RoboSim GUI has to be launched to tell the student's computer how to connect their code to the virtual robots. The default robot is a Linkbot-I matching the one to be controlled on line 2 of Program 1 at (0,0) with two wheels and a caster attached. This one button click will configure the computer to run the program and show the student a robot moving forward. Running the code through ChIDE will launch the RoboSim Scene as shown in Figure 7.

5.2 Controlling Two Robots in RoboSim

Extending what can be learned with one robot, two robots provide exponentially more options when exploring mathematic and science concepts. To control two or more robots with RoboSim, the steps are exactly the same as with one robot but repeated as necessary. Program 2 builds upon Program 1 to move two robots at different speeds to reach the same point in space.

The differences between controlling multiple robots and a single robot is as simple as adding new robots to the code (lines 3-4) and then calling the necessary functions for each of the different robots. Adding a second robot to the virtual environment through the RoboSim GUI is done through the configuration dialog which automatically adds the second robot next to the first one.

The code of Program 2 moves the two robots at two different speeds in the same direction for the same distance. Since the second robot moves at twice the speed of the first (line 11), it is delayed for half of the movement time so that the two robots end at the same instant.

```
#include <linkbot.h>

CLinkbotI robot1;
CLinkbotI robot2;

double speed = 5, distance = 10;
double delayTime = 1, radius = 1.75;

robot1.connect();
robot2.connect();
robot1.setTwoWheelRobotSpeed(speed);
robot2.setTwoWheelRobotSpeed(2*speed);
robot1.moveDistanceNB(distance, radius);
robot2.delaySeconds(delayTime);
robot2.moveDistance(distance, radius);
robot1.moveWait();
```

Program 2: Multiple robot control code.

There are numerous mathematical concepts which are laid out for the students within this program while still allowing them to calculate the mathematics by hand. The two `setTwoWheelRobotSpeed()` functions set the forward speed of the robots to be 5 and 10 units per second. The first robot is then moved for 10 units while the second robot waits for 1 second before itself moving 10 units. Since the first robot is moving at 5 units per second and moving 10 units, it will take a total of 2 seconds. The second robot will move twice as fast and therefore must delay for half of the total time. The end result of this program is an introduction to linear equations with robotics and the intersection of two lines. Taking the numbers from Program 2, the two equations for the motion are shown in Equations 1 and 2.

$$y = 5t \tag{1}$$

$$y = 10(t - 1) \tag{2}$$

Recording capabilities can be added to the program to generate a plot of the two robot's motion as shown in Figure 8 which is identical to the plot of the two linear equations.

5.3 Controlling Multiple Robot Modules in RoboSim

Beyond the simple use as a two-wheeled vehicle, the modular robots are able to be combined into larger configurations limited only by the builder. As part of RoboSim, a few complex shapes have been pre-built with which students can play. The explorer configuration, shown in Figure 9, consists of a wheeled vehicle with an arm and gripper attached to the top. The code to control the shape and move it around is more complicated but is provided for the students. These larger shapes are a starting point for further exploration into more advanced programming techniques.

Running the explorer code shows the explorer configuration moving around the virtual scene and moving its gripper to pick up an imaginary object which is shown in Figure 10.

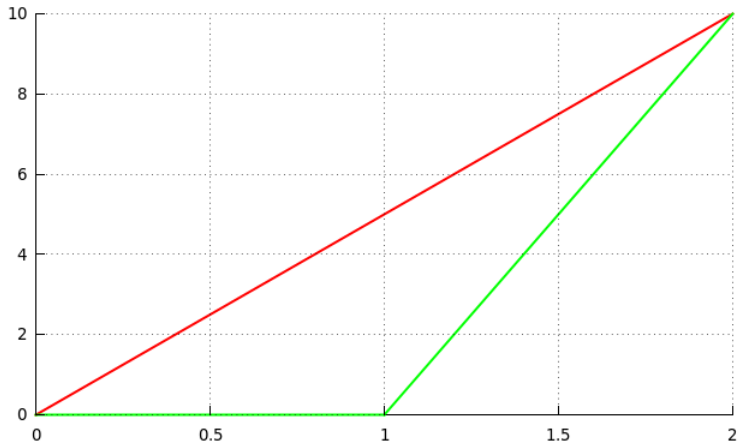

Figure 8: Plot of Two Robot motion.

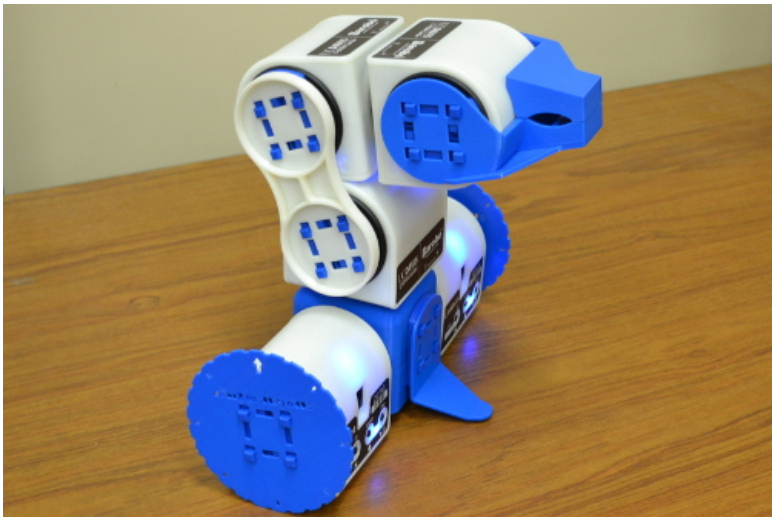

Figure 9: Explorer made out of Linkbot modules.

Being able to test student code developed to move the explorer from one location to another and pick up an object within simulation will save immense amounts of time when compared to having to test with the hardware robots. Each simulation can be started and restarted instantly as opposed to the hardware which has to be moved back into starting position each time. Once code has been developed which works within simulation, the code can be applied to the hardware robots without modification. The time required to create, test, and perfect a code for the explorer can be reduced by allowing students to quickly test many iterations before needing access to the hardware. Also, since the explorer is built with five Linkbot or three Mobot modules, having enough hardware for all students to have access at the same time is cost prohibitive. All students can have as many virtual robots as they want and just finish testing with the hardware when they have access.

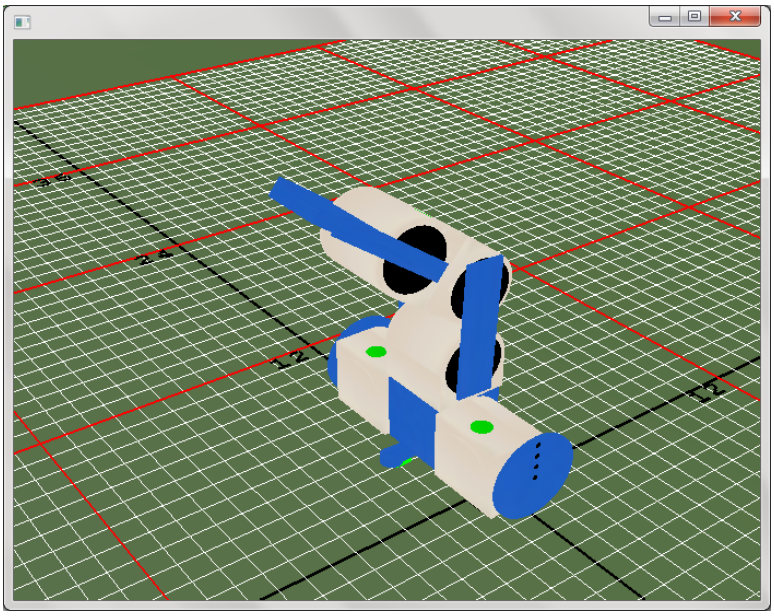

Figure 10: Virtual explorer made out of Linkbot modules.

6 Experience with RoboSim in Classroom Teaching

RoboSim has been tested within classrooms to validate it as an educational tool. Feedback from teachers is positive with regards to its intended goals. Data was collected from an even mix of middle and high school teachers teaching math, programming, and robotics classes. The greatest benefit expressed was the ease of use by not having to deal with connectivity and battery issues. While having hands-on robots is helpful for reinforcing the concepts, the hardware issues present when trying to deal with a classroom of thirty students is troublesome especially for a mathematics class. The simulation allows each student to work with his own robots which work all the time. Each student is also able to use as many robots as he wishes without running into cost limits as would happen with the physical robots.

Integration into the classroom is planned in a few different ways to help complement the hands-on robots. Introducing new topics to students and demonstrating how to use the robots for the new topic can be completed more easily through the simulation on the teacher's computer. As students are developing new code to run their robots, the simulation provides the platform to quickly alter the code before running on the hardware. When students are at home without robots, the simulation is being used to have homework assignments which integrate the robot lessons.

Initial data was collected from students who have been testing RoboSim within the classroom. Responses are positive and largely fell into a few categories. Since RoboSim is run purely on the computer, it is much easier for students to debug incorrectly behaving programs as it doesn't require the robots to be manually reset for each run. Once the virtual robots are behaving correctly, students can utilize the physical robots to verify the

simulation results.

Since the hardware robots have to deal with the inaccuracies present in the physical world, the results generated with RoboSim provide an error analysis check to what the students are seeing happen with the hardware. The discrepancy has led to discussions on the difference between theoretical calculations and actual results. Similar to the debugging benefits, RoboSim has also allowed students to quickly learn about the new functions by being able to quickly test and experiment to see how the new function behaves. Resetting the robot between motions is tedious when just trying to understand a function.

While only initial tests have been done in classrooms, the experience with RoboSim has been positive. The benefits are coming from the ease of use which provides more initial testing capabilities to complement the hardware robots hands-on learning benefits.

More testing is currently being completed which includes more widespread implementation in classrooms. Teachers and students are using RoboSim currently and will be able to give a more thorough analysis of its benefits soon.

7 Conclusions

Putting computer-based virtual robots into mathematics classrooms is able to supplement hands-on robotics projects to help reinforce mathematics concepts. RoboSim is designed and implemented to integrate with the hardware Linkbot and Mobot modular robots which have been designed to be used directly within the mathematics classrooms. While access to hardware cannot always be guaranteed, RoboSim allows students visualize the mathematics without needing the hardware. The code which has been written to control the virtual robots can be used without any modification to control the hardware robots for hands-on reinforcement. Initial classroom testing is showing that RoboSim has a place on the side of the hardware robots as a means to quickly test new motions before implementation on the physical robots. Combining the virtual robots of RoboSim and the physical robots, students are able to test, verify, and most importantly learn mathematics quickly and in an engaging way.

8 Acknowledgements

This work was supported in part by the National Science Foundation under Grant CNS-1132709, IIS-1208690, IIS-1256780, and IIP-1152678.

References

[1] California algebra 1 graduation requirements. http://www.cde.ca.gov/ci/gs/hs/algebrafaq.asp.

[2] David Silver, Marisa Saunders, and Estela Zarate. What factors predict high school graduation in the los unified school district. Technical report, California Dropout Research Project, June 2008.

[3] Consortium of Social Science Associations COSSA. Enhancing diversity in science: A leadership retreat on the role of professional associations and scientific societies: A summary report. Technical report, COSSA, 2008.

[4] T. Bredderman. What research says: Activity science-the evidence shows it matters. *Science and Children*, 20(1):39–41, 1982.

[5] Floyd Mattheis and Genzo Nakayama. Effects of a laboratory-centered inquiry program on laboratory skills, science process skills, and understanding of science knowledge in middle grades students. Technical report, Science and Mathematics Education Center; East Carolina University, 1988.

[6] F.J. Rutherford. Hands-on: A means to an end. *2061 Today*, 3(1):5, 1993.

[7] Gazebo. http://playerstage.sourceforge.net/index.php?src=gazebo.

[8] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[9] Marc Freese, Surya Singh, Fumio Ozaki, and Nobuto Matsuhira. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *Proceedings of the Second international conference on Simulation, modeling, and programming for autonomous robots*, SIMPAR'10, pages 51–62, Berlin, Heidelberg, 2010. Springer-Verlag.

[10] Iman Awaad and Beatriz Len. Xpersim: A simulator for robot learning by experimentation. In Stefano Carpin, Itsuki Noda, Enrico Pagello, Monica Reggiani, and Oskar Stryk, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 5325 of *Lecture Notes in Computer Science*, pages 5–16. Springer Berlin Heidelberg, 2008.

[11] Keyan Zahedi, Arndt von Twickel, and Frank Pasemann. Yars: A physical 3d simulator for evolving controllers for real robots. In Stefano Carpin, Itsuki Noda, Enrico Pagello, Monica Reggiani, and Oskar von Stryk, editors, *SIMPAR*, volume 5325 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 2008.

[12] Marilou. http://www.anykode.com/marilou.php.

[13] Robot virtual worlds. http://www.robotvirtualworlds.com.

[14] A. Liu, J. Newsom, C. Schunn, and R. Shoop. Students learn programming faster through robotic simulation. *Tech Directions*, pages 16–19, March 2013.

[15] Barobo, inc. http://www.barobo.com.

[16] G. Ryland and H. Cheng. Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion. *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 60–65, May 2010.

[17] Graham Ryland and Harry H Cheng. Novel Locomotion of iMobot, an Intelligent Reconfigurable Mobile Robot. In *Video Proc. of 2010 IEEE International Conference on Robotics and Automation*, May 2010.

[18] *Ch — an Embeddable C/C++ Interpreter*. http://www.softintegration.com (last modified April 15, 2009).

[19] David Ko, Harry H. Cheng, and Graham G. Ryland. Reconfigurable software for reconfigurable modular robots. In *Workshop on Modular Robotics: State of the Art, 2010 IEEE International Conference on Robotics and Automation*, 2010.

[20] ISO/IEC, Geneva, Switzerland. *International Standard: Programming languages - C*, 1990.

[21] ISO/IEC, Geneva, Switzerland. *International Standard: Programming languages - C*, 1999.

[22] Harry H Cheng. Handling of Complex Numbers in the Ch Programming Language. *Scientific Programming*, 2(3):76–106, Fall 1993.

[23] H.H. Cheng. Scientific Computing in the Ch Programming Language. *Scientific Programming*, 2(3):49–75, Fall 1993.